UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
http://www.icmc.usp.br
Departamento de Ciências de Computação

SCC-505 - Capítulo 1
Linguagens Regulares e Autômatos Finitos

João Luís Garcia Rosa[1]

[1]Departamento de Ciências de Computação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - São Carlos
http://www.icmc.usp.br/~joaoluis

2010

Sumário

Sumário

Um Exemplo: Parênteses Casados

- *Parênteses casados* incluem todas as cadeias balanceadas de parênteses esquerdos e direitos - por exemplo, (()), ()() e ((())()). A seguinte especificação descreve a linguagem de parênteses casados intuitivamente:
  1. A cadeia () é bem formada.
  2. Se a cadeia de símbolos A é bem formada, então a cadeia (A) também é.
  3. Se as cadeias A e B são bem formadas, então a cadeia AB também é.
- Pode-se agora seguir esta definição intuitiva para obter um sistema de re-escrita - uma **gramática** - que gera exatamente o conjunto de cadeias legais de parênteses casados:
  1. $S \rightarrow ()$
  2. $S \rightarrow (S)$
  3. $S \rightarrow SS$

Um Exemplo: Conceitos

- Estas três regras de re-escrita são chamadas de **produções**. Elas dizem "dada uma cadeia, pode-se formar uma nova cadeia substituindo um S por $()$ ou (S) ou SS". Para gerar $(())(()())$, ocorrem as seguintes substituições:

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow (())S \Rightarrow (())(S) \Rightarrow (())(SS) \Rightarrow (())(()S) \Rightarrow (())(()())$$

- Pode-se resumir a derivação anterior com a notação:

$$S \Rightarrow^* (())(()())$$

- que significa que "S deriva (em muitos passos) $(())(()())$". Quando se deseja descrever $v \Rightarrow w$ em palavras, onde v e w são cadeias arbitrárias, deve-se dizer que "v deriva diretamente w." Então SS deriva diretamente $(S)S$.

Um Exemplo: Conceitos

- A gramática apresenta dois tipos diferentes de símbolos: os **não-terminais**, ou variáveis, que são os símbolos que podem aparecer em derivações mas não aparecem nas cadeias finais (no exemplo, S é a única variável); e os **terminais**, que são os símbolos que formam a cadeia que se deseja produzir (no exemplo, “(” e “)” são símbolos terminais).

Sumário

Alfabeto e Linguagem

- Seja Σ um conjunto finito não vazio de símbolos terminais (**tokens**), ou **alfabeto terminal**. Seja Σ^* o conjunto de todas as cadeias de comprimento finito sobre Σ. Este conjunto infinito de cadeias inclui a cadeia vazia λ.
- Uma linguagem L sobre Σ é qualquer subconjunto de Σ^*. Então a linguagem de parênteses casados, M, é uma linguagem porque $M \subseteq \{(,)\}^*$. Cada cadeia w em L tem um comprimento finito, $|w|$, com $|\lambda| = 0$. Se $w = x_1 x_2 ... x_n$, cada $x_j \in \Sigma$, temos $|w| = n$. Podemos escrever Σ^+ para denotar a linguagem de cadeias não vazias sobre Σ. Se $w = x_1 x_2 ... x_n$ e $w' = x'_1 x'_2 ... x'_m$ são duas cadeias quaisquer, podemos concatená-las para obter
$$ww' = x_1 x_2 ... x_n x'_1 x'_2 ... x'_m$$
- Estabelece-se que $w\lambda = \lambda w = w$.

Gramática

- **Definição**. Uma **gramática de estrutura de frase** G é uma quádrupla (Σ, V, S, P) onde Σ e V são alfabetos finitos disjuntos e
  1. Σ é o alfabeto terminal para a gramática;
  2. V é o alfabeto não terminal ou alfabeto de variável para a gramática;
  3. S é o símbolo inicial para a gramática; e
  4. P é conjunto de produções da gramática. P é um conjunto de pares (v, w) com v uma cadeia em $(\Sigma \cup V)$ contendo no mínimo um símbolo não terminal, enquanto que w é um elemento arbitrário de $(\Sigma \cup V)^*$. Um elemento (v, w) de P é usualmente escrito como $v \rightarrow w$.
- A gramática para a linguagem de parênteses casados pode ser escrita como

$$G = (\{(,)\}, \{S\}, S, \{S \rightarrow (), S \rightarrow (S), S \rightarrow SS\})$$

Linguagem

- **Definição**. Seja G uma gramática com símbolo inicial S. A **linguagem** gerada por G, $L(G)$, é definida como o conjunto de cadeias terminais deriváveis do símbolo inicial:

$$L(G) = \{w | w \in \Sigma^* \text{ e } S \Rightarrow^* w\}$$

- Dado G, se $S \Rightarrow^* w$ (onde w é em geral construído tanto de símbolos terminais como não terminais) referimos a w como uma **forma sentencial**. Então $L(G)$ é o conjunto de formas sentenciais terminais.

Exemplos

- **Exemplo 1**: Sejam $V = \{S\}$, $\Sigma = \{a, b\}$, $P = \{S \to aSb, S \to ab\}$. Neste caso $L(G) = \{a^n b^n \mid n > 0\}$. Ou seja, $L(G)$ é o conjunto de todas as seguintes cadeias: um bloco não vazio de a's, seguido por um bloco de b's do mesmo comprimento.
- **Exemplo 2**: O conjunto de produções de uma gramática
$$S \to aSa \mid bSb \mid a \mid b \mid \lambda$$
gera o conjunto de todos os palíndromos sobre o alfabeto terminal $\{a, b\}$.
- **Exemplo 3**: A seguinte gramática gera todas as cadeias sobre o alfabeto terminal $\{0, 1\}$ com um número igual de 0's e 1's:
  - $V = \{S, A, B\}$
  - $\Sigma = \{0, 1\}$
  - $P = \{S \to 0B \mid 1A,\ A \to 0 \mid 0S \mid 1AA,\ B \to 1 \mid 1S \mid 0BB\}$.

Gramática Linear a Direita - GLD

- **Definição**: Uma gramática $G = (\Sigma, V, S, P)$ é **linear a direita** (GLD) se toda produção é da forma

$$A \rightarrow bC \text{ ou } A \rightarrow b$$

onde $A, C \in V$ e $b \in \Sigma \cup \{\lambda\}$. Uma linguagem gerada por tal gramática é chamada de linguagem linear a direita.

- **Exemplo 4**: Considere a GLD G_1 com produções

$$S \rightarrow \lambda \mid 0 \mid 0S \mid 1 \mid 1S$$

Claramente, $L(G_1) = \{0, 1\}^*$, o conjunto de todas as cadeias binárias.

- **Exemplo 5**: Agora, considere a GLD G_2:

$$S \rightarrow \lambda \mid 0S \mid 1T,$$
$$T \rightarrow 0T \mid 1S$$

Pode-se provar que

$$L(G_2) = \{w | w \in \{0, 1\}^* \text{ e } w \text{ tem um número par de 1's } \}$$

BNF

- O estilo de linguagem de programação para escrever gramáticas, a chamada **forma de Backus-Naur**, ou **BNF**, usa uma notação levemente diferente. Na BNF as variáveis são fechadas em $< >$, e $\rightarrow$ é substituído pelo símbolo "::=". Portanto, a segunda produção do exemplo dos parênteses casados:

$$S \rightarrow (S)$$

- é escrita na notação BNF como:

$$< S > ::= (< S >)$$

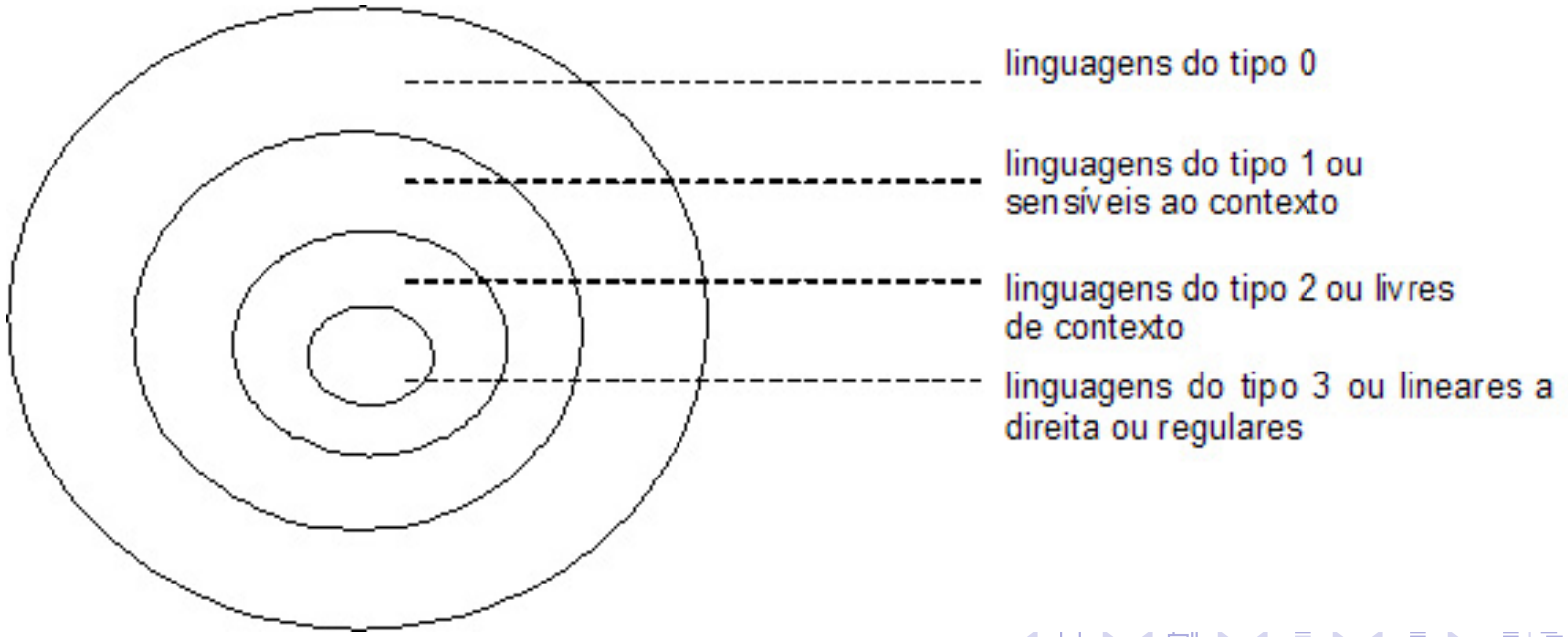
Tipos de Gramáticas e Linguagens

- **Definição**: Seja $G = (\Sigma, V, S, P)$ uma gramática. Então G é classificada como tipo i, $i = 0, 1, 2, 3$, de acordo com restrições na forma das regras de P:
  - Uma gramática é do **tipo 3** se toda produção de P for da forma $A \rightarrow bC$ ou $A \rightarrow b$, onde $A, C \in V$ e $b \in \Sigma$ ou $b = \lambda$. Tais gramáticas são as **gramáticas lineares a direita**.
  - Uma gramática é do **tipo 2** se toda produção de P for da forma $A \rightarrow w$, com $A \in V$ e $w \in (\Sigma \cup V)^*$. Estas são as **gramáticas livres de contexto**.
  - Uma gramática é do **tipo 1** se toda produção de P for da forma $vAw \rightarrow vzw$ onde $z \in (\Sigma \cup V)^+$ (isto é, $z \neq \lambda$). Em adição, permite-se a única regra-λ $S \rightarrow \lambda$ no caso onde S não aparece do lado direito de nenhuma produção. São as **gramáticas sensíveis ao contexto**.
  - Qualquer gramática é do **tipo 0**. Não existem restrições na forma das produções para as gramáticas desta classe. São chamadas de **gramáticas irrestritas**.

A Hierarquia de Chomsky

- Vai-se usar a notação $\mathcal{L}_i$ para as linguagens de tipo i:
  $\mathcal{L}_i = \{L \in \Sigma^* | L = L(G)$ para alguma gramática G de tipo $i\}$
- **O Teorema da Hierarquia**: A hierarquia de Chomsky é uma hierarquia estrita de classes de linguagem:

$$\mathcal{L}_3 \subsetneq \mathcal{L}_2 \subsetneq \mathcal{L}_1 \subsetneq \mathcal{L}_0$$



Sumário

Linguagem Regular

- **Definição**: Seja Σ um alfabeto finito. Uma linguagem $L \subseteq \Sigma^*$ é **regular** se L for finita, ou L pode ser obtida indutivamente usando uma das seguintes operações:
  1. L é $L_1 \cup L_2$, a união de L_1 e L_2, onde L_1 e L_2 são regulares; ou
  2. L é $L_1 \cdot L_2$, a concatenação de L_1 e L_2, onde L_1 e L_2 são regulares; ou
  3. L é L_1^*, a iteração de L_1, onde L_1 é regular.
- Note que o conjunto vazio $\emptyset$ e o conjunto $\{\lambda\}$ são ambos regulares, sendo finitos. As linguagens
  1. $\{ab\}^* \cdot \{a\}^*$
  2. $\{a, b\}^*$

  são também regulares. Por que?

Expressão Regular

- **Proposição**: Todo conjunto regular é uma linguagem linear a direita.
- **Definição**: Seja Σ um alfabeto finito. Define-se **expressões regulares** R e suas denotações de conjuntos regulares $S(R)$ indutivamente como:
  1. $\emptyset$ é uma expressão regular com $S(\emptyset) = \emptyset$, o conjunto vazio.
  2. λ é uma expressão regular com $S(\lambda) = \{\lambda\}$.
  3. Se $a \in \Sigma$, então a é uma expressão regular com $S(a) = \{a\}$.
  4. Se R_1 e R_2 são expressões então $(R_1 + R_2)$ ou $(R_1|R_2)$ é uma expressão regular com $S((R_1 + R_2)) = S((R_1|R_2)) = S(R_1) \cup S(R_2)$.
  5. Se R_1 e R_2 são expressões então $(R_1 \cdot R_2)$ é uma expressão regular com $S((R_1 \cdot R_2)) = S(R_1) \cdot S(R_2)$.
  6. Se R é regular então $(R)^*$ é regular com $S((R)^*) = (S(R))^*$.

Expressão Regular: Exemplos

1. $(a + b)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$
2. $(a^* \cdot b^*)^*$ denota todas as cadeias sobre $\Sigma = \{a, b\}$ (Por que?)
3. $(aa + bb)^*$, que abrevia $((a \cdot a) + (b \cdot b))^*$, representa todas as cadeias finitas de a's e b's construídas pela concatenação de pares de a's e pares de b's.
4. $aa(b^* + aaa)c + (a + c)^*$ representa a linguagem que é a união de três sublinguagens:
   1. todas as cadeias começando com um a duplo, seguido por qualquer número de b's, seguido por um c;
   2. a linguagem cadeia $\{aaaaac\}$; e
   3. a linguagem construída por todas as cadeias de a's e c's.

Expressão Regular: Extensões

- Objetivo das extensões às expressões regulares de Kleene:
  - melhorar a capacidade de especificar os padrões de entrada.
- Algumas extensões (usadas inicialmente no *Lex* - utilitário Unix), úteis na especificação de analisadores léxicos:
  1. *Uma ou mais instâncias*: operador $^+$. Leis algébricas: $r^* = r^+ \mid \lambda$, $r^+ = rr^* = r^*r$
  2. *Zero ou uma instância*: operador ?. Lei algébrica: $r? = r \mid \lambda$, ou $L(r?) = L(r) \cup \{\lambda\}$.
  3. *Classes de caracteres*: a expressão regular $a_1 + a_2 + ... + a_n$, onde $a_i \in \Sigma$, pode ser substituída por $[a_1 a_2 ... a_n]$. Quando $a_1 a_2 ... a_n$ formam uma sequência lógica, pode-se substituí-la por $a_1 - a_n$. Logo, $[abc] = a + b + c$, e $[a - z] = a + b + ... + z$.

Expressão Regular: Extensões

- Exemplos:
  1. Identificadores da linguagem C no padrão de Kleene:
     - $letra\_ \rightarrow A \mid B \mid \cdots \mid Z \mid a \mid b \mid \cdots \mid z \mid \_$
     - $digito \rightarrow 0 \mid 1 \mid \cdots \mid 9$
     - $id \rightarrow letra\_(letra\_ \mid digito)^*$
  2. Identificadores da linguagem C no padrão estendido:
     - $letra\_ \rightarrow [A - Zaz\_]$
     - $digito \rightarrow [0 - 9]$
     - $id \rightarrow letra\_(letra\_ \mid digito)^*$

Expressão Regular e Gramática

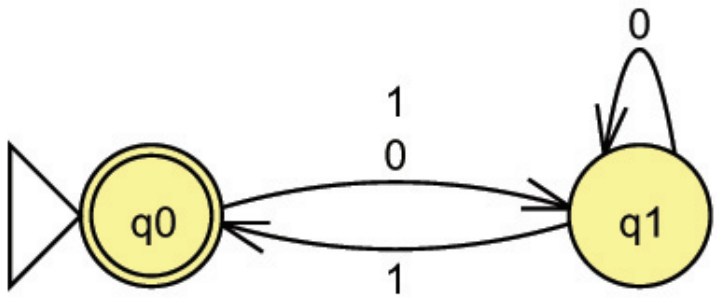
- **Exemplo**: Considere a expressão regular $(b^* + (ab)^*)$. Para achar uma gramática linear a direita para esta expressão, primeiro forme uma gramática para b^*: $S_1 \rightarrow bS_1|\lambda$. Então forme uma gramática para (ab): $S_2 \rightarrow aB_2, B_2 \rightarrow b$. Isto pode ser estrelado adicionando $S_2 \rightarrow \lambda$, $B_2 \rightarrow bS_2$. Finalmente, a linguagem completa pode ser gerada adicionando $S \rightarrow bS_1|\lambda|aB_2$, levando a

$$\begin{aligned} S &\rightarrow bS_1|\lambda|aB_2 \\ S_1 &\rightarrow bS_1|\lambda \\ S_2 &\rightarrow aB_2|\lambda \\ B_2 &\rightarrow bS_2|b \end{aligned}$$

Sumário

Autômato Finito

- Nesta seção, introduzimos a classe básica de dispositivos de computação chamados de **autômatos finitos**. Estes dispositivos julgam a legalidade de cadeias sobre algum alfabeto finito. Vamos mostrar que a coleção de linguagens que podem ser aceitas por autômatos finitos é exatamente a de linguagens regulares.
- **Exemplo**. O autômato finito dado a seguir aceita a linguagem $((0 + 1)0^*1)^*$.

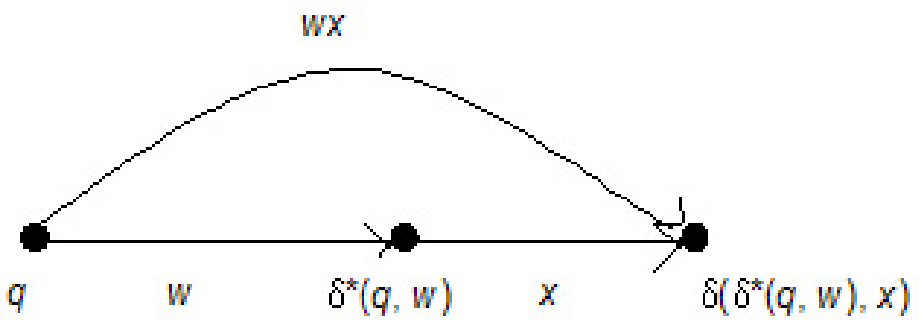


Autômato Finito Determinístico

- **Definição**: Um **autômato finito determinístico** (AFD) M é especificado por uma quíntupla $(Q, \Sigma, \delta, q_0, F)$ onde
  - Q é um conjunto finito de estados
  - Σ é o alfabeto terminal
  - δ: $Q \times \Sigma \rightarrow Q$ é a função de transmissão de estado
  - $q_0 \in Q$ é o estado inicial
  - $F \subset Q$ é o conjunto de estados de aceitação.
- Então o **grafo de estado** de um AFD tem um nó para cada $q \in Q$, que é rotulado; tem uma seta livre apontando para o nó q_0; tem um círculo duplo para um nó q apenas no caso de $q \in F$; e tem um arco rotulado x levando de um nó q para um nó q' apenas no caso de $\delta(q, x) = q'$.

Autômato Finito Determinístico

- **Definição**: Dado um AFD M, define-se por indução a função δ^*: $Q \times \Sigma^* \rightarrow Q$, tal que $\delta^*(q, w)$ é o estado para o qual M irá na cadeia de entrada w se começado no estado q_0:
  - Passo Básico: $\delta^*(q, \lambda) = q$ para cada estado $q \in Q$. Se M começou no estado q, então quando ele receber a cadeia de entrada vazia ele deve ainda estar no mesmo estado q.
  - Passo de Indução: Para cada estado $q \in Q$, cada cadeia de entrada $w \in \Sigma^*$ e cada símbolo de entrada $x \in \Sigma$, estabelece-se que $\delta^*(q, wx) = \delta(\delta^*(q, w), x)$.



Linguagem de Estados Finitos

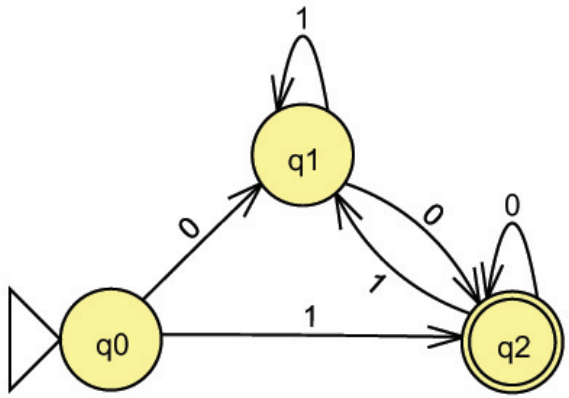
- **Definição**: O conjunto $T(M)$ de cadeias aceitas pelo AFD $M = (Q, \Sigma, \delta, q_0, F)$ é

$$T(M) = \{w | w \in \Sigma^* \text{ e } \delta^*(q_0, w) \in F\}$$

  compreendendo todas as cadeias terminais que enviam M do seu estado inicial q_0 para um estado de aceitação, isto é, um estado em F.
- Diz-se que um subconjunto L de Σ^* é uma **linguagem de estados finitos** (LEF) se for igual a $T(M)$ para algum AFD M.
- **Proposição**: Toda LEF é uma linguagem linear a direita.
- **Teorema**: Toda LEF é um conjunto regular.

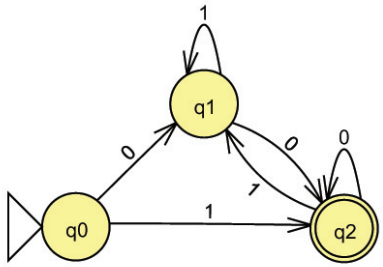
Expressão Regular Equivalente

- Seja o seguinte autômato:



- Para gerar a expressão regular equivalente a este autômato, é necessário estabelecer todos os caminhos que levam do estado inicial q_0 a um estado de aceitação (neste caso q_2). Os caminhos são:
  - De q_0 a q_2: $1 \cdot 0^*$
  - De q_0 a q_1 a q_2: $0 \cdot 1^* \cdot 0 \cdot 0^*$
  - De q_0 a q_1 a q_2 a q_1 a q_2: $0 \cdot 1^* \cdot 0 \cdot 0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*$

Expressão Regular Equivalente



- Observe que há um ciclo se repetindo entre q_2 e q_1, logo pode-se representar os caminhos que levam de q_0 a q_2 passando por q_1 da seguinte forma:
  - De q_0 a q_1 a q_2 a $(q_1$ a $q_2)^*$: $0 \cdot 1^* \cdot 0 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$
- Outro caminho é de q_0 a q_2 e realizando o ciclo $q_1 \Leftrightarrow q_2$:
  - De q_0 a q_2 a $(q_1$ a $q_2)^*$: $1 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$, que inclui $1 \cdot 0^*$ inicial.
- Logo a expressão total fica:

$$1 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^* + 0 \cdot 1^* \cdot 0 \cdot 0^* \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^* = (1 \cdot 0^* + 0 \cdot 1^* \cdot 0 \cdot 0^*) \cdot (0^* \cdot 1 \cdot 1^* \cdot 0 \cdot 0^*)^*$$

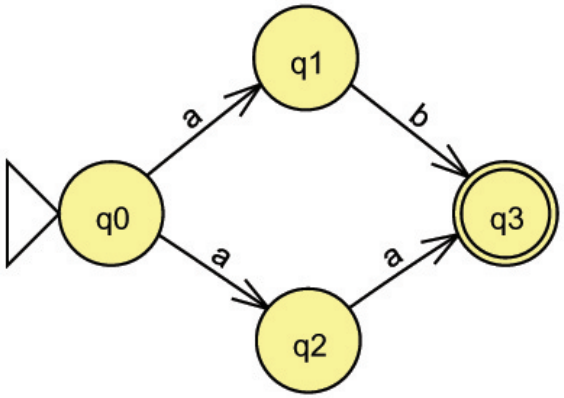
Sumário

Não Determinismo

- Deseja-se provar que toda linguagem linear a direita é uma LEF. Considere a seguinte gramática linear a direita:

$$(\{a, b\}, \{q_0, q_1, q_2\}, q_0,$$
$$\{q_0 \rightarrow aq_1, q_1 \rightarrow b, q_0 \rightarrow aq_2, q_2 \rightarrow a\})$$

- Esta gramática leva à seguinte máquina:



- A estrutura obtida não é um AFD, porque dois arcos saindo de q_0 têm o mesmo rótulo, a. Isto viola a condição da regra de transição para um AFD ser uma função.

Autômato Finito Não Determinístico

- Quando transições múltiplas deste tipo estão presentes em M, diz-se que M é uma **máquina não-determinística**.
- **Definição**: Um **autômato finito não determinístico** (AFN) M é especificado por uma quíntupla $(Q, \Sigma, \delta, Q_0, F)$ onde
  - Q é um conjunto finito de estados
  - Σ é o alfabeto terminal
  - δ: $Q \times \Sigma \rightarrow 2^Q$ atribui a cada par estado-entrada (q, x) um conjunto $\delta(q, x) \subset Q$ de próximos estados possíveis (2^Q é o conjunto potência de Q ou conjunto de todos os subconjuntos de Q)
  - $Q_0 \subset Q$ é o conjunto de estados iniciais
  - $F \subset Q$ é o conjunto de estados de aceitação.

Autômato Finito Não Determinístico

- Viu-se que um AFN é como um AFD exceto que existe um conjunto de estados iniciais e um conjunto de próximos estados possíveis.
- Estas complicações significam que uma cadeia pode induzir caminhos múltiplos através de um AFN, alguns terminando em estados de aceitação, outros terminando em estados de não aceitação.
- Lida-se com esta ambiguidade dizendo que $w \in \Sigma^*$ é aceita se existe pelo menos uma forma de chegar a um estado de aceitação.
- Vai-se definir agora $\delta^* : 2^Q \times \Sigma^* \rightarrow 2^Q$ para um AFN tal que $\delta^*(P, w)$ é o conjunto de estados alcançáveis a partir de algum estado em P pela aplicação da cadeia de entrada w.

Autômato Finito Não Determinístico

- **Definição**: Estenda $\delta : Q \times \Sigma \rightarrow 2^Q$ para um AFN para $\delta^* : 2^Q \times \Sigma^* \rightarrow 2^Q$ por:
  - Passo Básico: $\delta^*(P, \lambda) = P$ para cada $P \subset Q$.
  - Passo de Indução: $\delta^*(P, wx) = \bigcup\{\delta(q, x) | q \in \delta^*(P, w)\}$ para cada $w \in \Sigma^*$, $x \in \Sigma$ e $P \subset Q$.
- Estabelece-se que

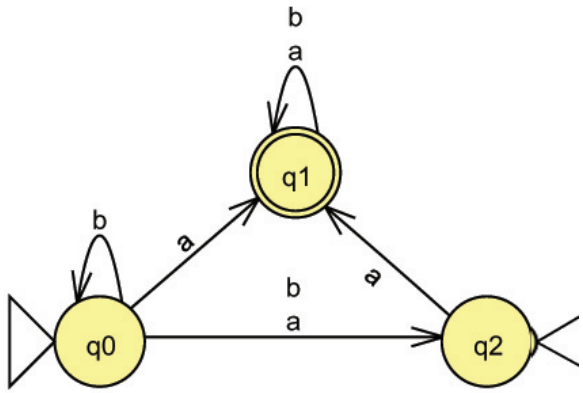
$$T(M) = \{w | w \in \Sigma^* \text{ e } \delta^*(Q_0, w) \cap F \neq \emptyset\}$$

é o conjunto de cadeias de entrada tal que no mínimo um caminho, rotulado com símbolos de w em ordem, através do grafo de estado leva M de um estado inicial para um estado de aceitação.

- **Proposição**: Um subconjunto de Σ^* é uma linguagem de estados finitos se e somente se ele for $T(M)$ para algum AFN M.

Conversão de AFN para AFD

- **Exemplo**: Considere o seguinte AFN.



- Os estados q_0 e q_2 são estados iniciais, enquanto que o estado q_1 é o único estado de aceitação.
- Existem $2^{|Q|}$ (neste caso, $2^3 = 8$) estados no AFD equivalente, mas na prática não é necessário considerar todos eles, porque muitos são inalcançáveis.

Conversão de AFN para AFD

- Tira-se vantagem desta observação começando do novo estado inicial q_{02}, correspondente à união de q_0 e q_2, e construindo estados adicionais na medida em que eles são gerados. Começa-se com

δ	a	b
$\{q_0, q_2\}$	$\{q_2, q_1\}$	$\{q_0, q_2\}$

que descreve a ação de δ no estado inicial q_{02} de entradas a e b, respectivamente. De maneira análoga, chamar-se-á o estado $\{q_2, q_1\}$ de q_{21}:

δ	a	b
q_{02}	q_{21}	q_{02}

Conversão de AFN para AFD

- Apenas um novo estado foi gerado, assim produz-se sua linha:

δ	a	b
q_{21}	q_1	q_1

- Então tem-se:

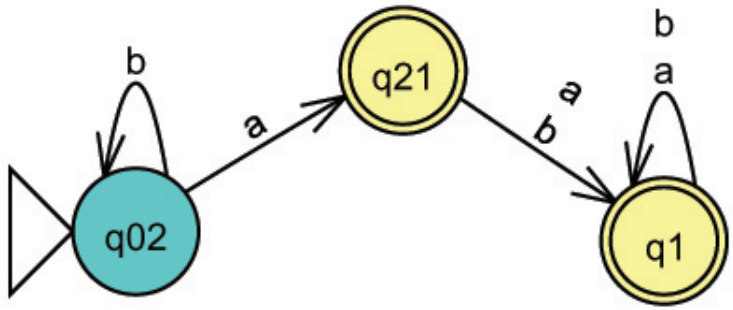
δ	a	b
q_1	q_1	q_1

- A tabela completa do δ fica:

δ	a	b
q_{02}	q_{21}	q_{02}
q_{21}	q_1	q_1
q_1	q_1	q_1

Conversão de AFN para AFD

- Todos os estados resultantes que contêm estados finais do AFN original são estados de aceitação no AFD correspondente. O diagrama de estados do AFD equivalente é:



- **Teorema**: Toda linguagem linear a direita *L* é uma LEF.
- Conclui-se portanto que as classes de linguagens lineares a direita, linguagens regulares e linguagens de estados finitos coincidem.

Referências I

[1] Hopcroft, J. E., Ullman, J. D.
Formal Languages and Their Relation to Automata.
Addison-Wesley Publishing Company, 1969.

[2] Hopcroft, J. E., Ullman, J. D. e Motwani, R.
Introdução à Teoria de Autômatos, Linguagens e Computação.
Tradução da segunda edição americana. Editora Campus, 2003.

[3] Menezes, P. B.
Linguagens Formais e Autômatos.
Série Livros Didáticos. 4a. Edição. Instituto de Informática da UFRGS. Editora Sagra Luzzatto, 1997.

Referências II


[4] Moll, R. N., Arbib, M. A., and Kfoury, A. J.
An Introduction to Formal Language Theory.
Springer-Verlag, 1988.


[5] Rosa, J. L. G.
Teoria da Computação e Linguagens Formais.
Slides. SCE0205. Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, 2009.